Dissecting a RAT. Analysis of the Command-line AndroRAT.

stratosphereips.org/blog/2021/5/6/dissecting-a-rat-analysis-of-the-command-line-androrat

Kamila Babayeva June 1, 2021

This blog post was authored by Kamila Babayeva (@_kamifai_) and Sebastian Garcia (@eldracote).

The RAT analysis research is part of the Civilsphere Project (https://www.civilsphereproject.org/), which aims to protect the civil society at risk by understanding how the attacks work and how we can stop them. Check the webpage for more information.

This is the seventh blog of a series analyzing the network traffic of Android RATs from our Android Mischief Dataset [more information here], a dataset of network traffic from Android phones infected with Remote Access Trojans (RAT). In this blog post we provide the analysis of the network traffic of the RAT08-command-line-AndroRAT [download here]. The previous blogs analyzed Android Tester RAT, DroidJak RAT, AndroRAT RAT, SpyMax RAT, AhMyth RAT and HawkShaw RAT.

Execution Setup

The goal of each of our RAT experiments is to use the software ourselves and to execute every possible action while capturing all the traffic and storing all the logs. So these RAT captures are functional and were used in real attacks.

Despite its name “Command line AndroRAT”, this RAT has no clear relationship with the RAT called “AndroRAT”. The Command line AndroRAT is a software package that contains the controller software and builder software to build an APK. It was executed on a Windows 7 guest virtual machine with Ubuntu 20.04 as a host. The Android Application Package (APK) built by the RAT builder was installed in the Android virtual emulator called Genymotion using Android version 8.

While performing different actions on the RAT controller (e.g. upload a file, get GPS location, monitor files, etc.), we captured the network traffic on the Android virtual emulator.

The details about the network traffic capture are:

- The controller IP address: 147.32.83.157
- The phone IP address: 147.32.83.245

- UTC time of the infection in the capture: 2020-12-05 11:46:43 UTC

RAT Details

This Command-line AndroRAT software was the first one in our dataset that did not have an graphical user interface. Instead, it uses a command line interface to control the target's device. Figure 1 shows the welcome message in the command line while waiting for the infected device to connect.This Command-line AndroRAT software was the first one in our dataset that did not have an graphical user interface. Instead, it uses a command line interface to control the target's device. Figure 1 shows the welcome message in the command line while waiting for the infected device to connect.

Figure 1. Welcome message in the Command-line AndroRAT interface. The message is shown until the infected phone is connected.

Infection and Initial Communication

This research started with the execution of the RAT in our virtual phone. Once the APK was installed in the phone, it directly tried to establish a TCP connection with the command and control (C&C) server. The phone used the IP address and the port of the controller that we specified in the APK (Figure 2). In particular, the IP address of the controller was 147.32.83.157 and the port was 1337/TCP. The controller IP address 147.32.83.157 is the IP address of a Windows 7 virtual machine in our lab computer, meaning that the IP address is not connected to any indicator of compromise (IoC).

```
public class config {

    /* renamed from: IP */
    public static String f33IP = "147.32.83.157";
    public static String port = "1337";
}
```

Figure 2. The controller IP and port specified during compilation can be seen in the code inside the APK installed in the victim's device. The phone uses the controller IP 147.32.83.157 and the port 1337 to establish a TCP connection.

The phone initializes a 3-way TCP handshake to establish the connection between the phone and the C&C. Figure 3 shows these initial packets. The connection was successfully established without any reconnections, but with a retransmission packet. The lack of reconnections can be because both controller and victim were in the same network.

1431	2020-12-05 11:46:43,772135	147.32.83.245	46032	147.32.83.157	1337	TCP	74 46032 → 1337 [SYN] Seq=0
1432	2020-12-05 11:46:43,773043	147.32.83.157	1337	147.32.83.245	46032	TCP	74 1337 → 46032 [SYN, ACK] S
1433	2020-12-05 11:46:43,773342	147.32.83.245	46032	147.32.83.157	1337	TCP	66 46032 → 1337 [ACK] Seq=1
1434	2020-12-05 11:46:43,776767	147.32.83.245	46032	147.32.83.157	1337	TCP	117 46032 → 1337 [PSH, ACK] S
1435	2020-12-05 11:46:43,989669	147.32.83.245	46032	147.32.83.157	1337	TCP	117 [TCP Retransmission] 460
1436	2020-12-05 11:46:43,990173	147.32.83.157	1337	147.32.83.245	46032	TCP	78 1337 → 46032 [ACK] Seq=1

Figure 3. A 3-way TCP handshake between the controller (147.32.83.157) and the phone (147.32.83.245). The connection was initialized by the phone and there is one retransmission packet.

After the phone got infected and the connection between the phone and the controller was established, the phone sent a welcome message together with the phone model "Samsung-2", as shown in Figure 4. The code from the malicious APK that sends the welcome message to the controller is shown in Figure 5.

```
0000   48 65 6c 6c 6f 20 74 68 65 72 65 2c 20 77 65 6c   Hello there, wel

0010   63 6f 6d 65 20 74 6f 20 72 65 76 65 72 73 65 20   come to reverse

0020   73 68 65 6c 6c 20 6f 66 20 53 61 6d 73 75 6e 67   shell of Samsung

0030   2d 32 0a                                          -2.
```

Figure 4. The welcome message with the model of the phone sent from the infected phone to the controller after a successful infection. Notice the English language

```
String welcomeMess = "Hello there, welcome to reverse shell of " + (Build.MODEL + IOUtils.LINE_SEPARATOR_UNIX);
```

Figure 5. Code from the malicious APK that sends the welcome message to the C&C.

After sending a welcome message, the phone waits for the C&C command. While waiting for the C&C command, there was no heartbeat performed between the phone and the controller.

The phone then received its first executed C&C command ‘device info’ that aims to retrieve the details about the phone’s hardware, system, settings, etc. Figure 6 shows the data field of the packet with the command ‘device info’. The C&C command is sent in the plain text, without any structure.

```
0000   64 65 76 69 63 65 49 6e 66 6f 0a                deviceInfo.
```

Figure 6. The data field of the packet with the C&C command ‘device info’ that aims to retrieve the details about the infected device. The data is in the plain text without any structure.

The phone answers to the command ‘device info’ with device details composed of Manufacturer, Version/Release, Product, Model, Brand, Device and Host. The data field of this packet is displayed in Figure 7. It is important to notice that the answer from the phone does not follow any structure, the data is sent in the plain text.

```
----------------------------------------

Manufacturer: unknown

Version/Release: 8.1.0

Product: vbox86p

Model: Samsung-2

Brand: Android

Device: vbox86p

Host: 49cfa9ee5067

----------------------------------------
```

Figure 7. The data field of the packet with the phone’s answer to the C&C command ‘device info’. The data is sent in the plain without any structure. It may seem that the controller is separating these values by searching for the words “Manufacturer:”, “Version/Release”, etc.

The request and answer to the C&C command ‘device info’ are shown in the command line interface of the C&C, as shown in Figure 8.

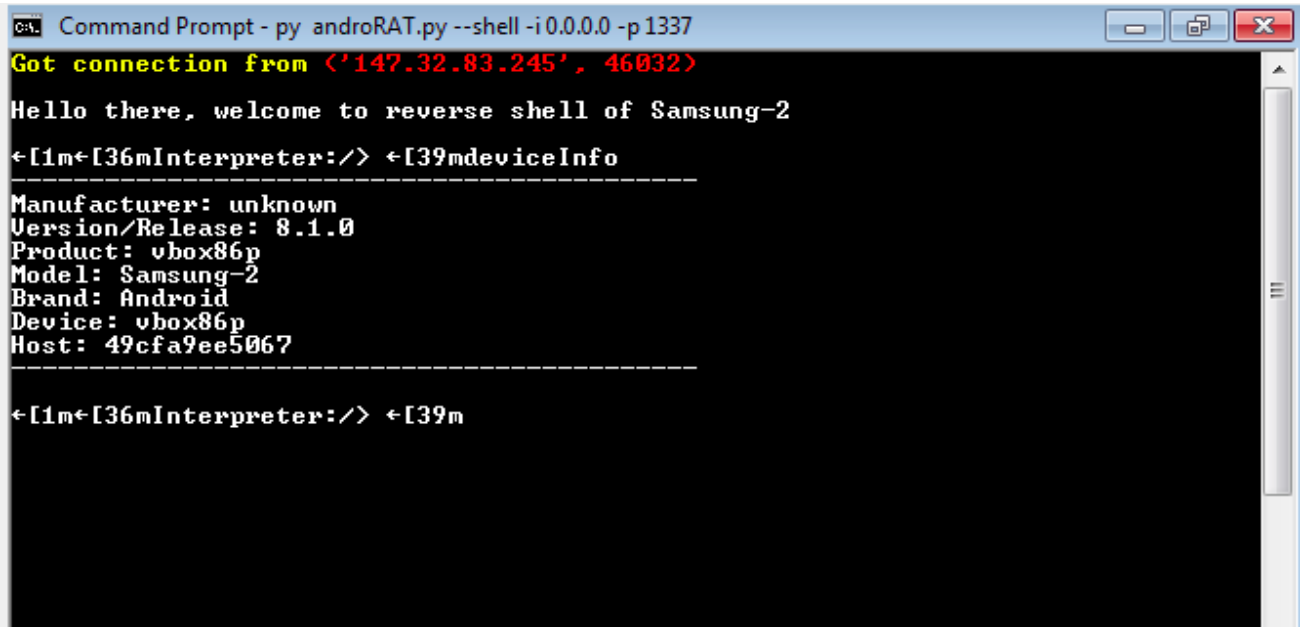

Figure 8. The command line interface of the C&C with the executed command 'Device Info' and the phone's reply. The characters "[36m" and similar seem to be related to a bug in the assignment of colors to the interface.

Example of C&C Commands

Through the whole communication, the controller sends the C&C commands inplaintext, the phone answers these commands in plaintext as well. When the controller or the victim sends a big amount of data, e.g. photo, video, audio, text files., it defines the end of data by adding a string 'END123\n' at the end.

As an example we can analyze the exchange of packets between the C&C and the victim during the C&C command 'getSMS'. This command aims to retrieve the messages sent and received by the targeted device. The data of the packet with the 'getSMS' command is displayed in Figure 9. As before, the data is sent in plaintext and does not follow any structure. As a reply to this command, the phone sends two packets: the first packet confirms the execution of the C&C command (Figure 10), the second packet sends the actual data (Figure 11).

```
0000   67 65 74 53 4d 53 20 69 6e 62 6f 78 0a        getSMS inbox.
```

Figure 9. The data field of the packet sent by the controller with C&C command 'getSMS' that aims to retrieve the message inbox inside the targeted phone.

```
0000   72 65 61 64 53 4d 53 20 69 6e 62 6f 78 0a     readSMS inbox.
```

Figure 10. The data field of the packet sent by the victim phone with the text 'readSMS' as a confirmation answer to the command "getSMS".

```
#0
Number : 333333
Person : null
Date : Sun Jun 13 13:18:52 EST 52877
Body : Hey! i am thwoing a party at my house next week! wanna join?

#1
Number : 928934
Person : null
Date : Sun Jun 13 04:14:21 EST 52877
Body : Hello! How are you and your child? Are you back from vacation already?


END123
```

Figure 11. The data field of the phone reply to the command 'getSMS'. The messages are sent in the plaintext. In order to define the end of the data, it puts the 'END123\n' string at the end of the data. The fields seem to be separated, again, by searching for keywords such as "Number", "Person", etc.

There are a total of 18 commands that the RAT software can perform on the targeted device. The complete list is shown in Figure 12.

```
deviceInfo                      --> returns basic info of the device
camList                         --> returns cameraID
takepic [cameraID]              --> Takes picture from camera
startVideo [cameraID]           --> starts recording the video
stopVideo                       --> stop recording the video and return the video
file
startAudio                      --> starts recording the audio
stopAudio                       --> stop recording the audio
getSMS [inbox|sent]             --> returns inbox sms or sent sms in a file
getCallLogs                     --> returns call logs in a file
shell                           --> starts a interactive shell of the device
vibrate [number_of_times]       --> vibrate the device number of time
getLocation                     --> return the current location of the device
getIP                           --> returns the ip of the device
getSimDetails                   --> returns the details of all sim of the device
clear                           --> clears the screen
getClipData                     --> return the current saved text from the
clipboard
getMACAddress                   --> returns the mac address of the device
exit                            --> exit the interpreter
```

Figure 12. The complete list of 18 commands that can be used from the controller. It is a print of the help function in the C&C interface.

End of communication

After the C&C sends the command ‘exit’ (Figure 13), the connection between the phone and the controller should have been closed. However, in our experiment, after the connection was closed (Figure 14), the phone attempts to reconnect to the C&C several times with an interval of 3 seconds (Figure 15), showing a buggy implementation of the exit function in the APK, or showing that the controller may no longer be active but giving the victims the opportunity to reconnect if necessary.

```
0000   65 78 69 74 0a  exit.
```

Figure 13. The C&C command ‘exit’ that aims to close the connection between the phone and the controller.

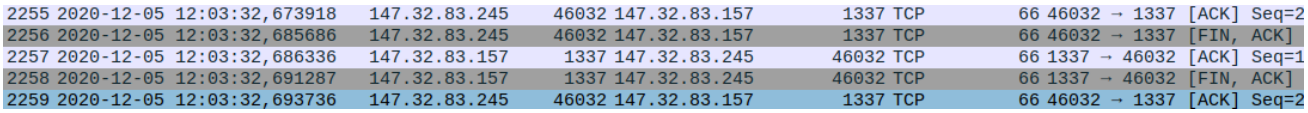

Figure 14. Successful 4-way handshake TCP termination between the controller and the targeted phone after the C&C command ‘exit’.

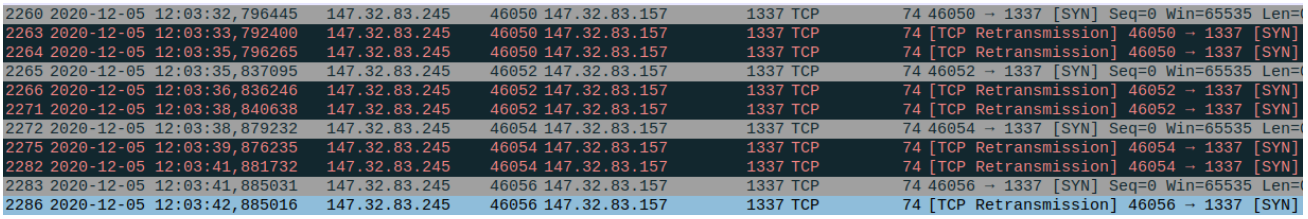

Figure 15. After the phone received the ‘exit’ C&C command, it still tries to reconnect with the controller. However, the controller already closed the socket after the ‘exit’ C&C command.

The complete communication between the phone and the controller in the experiment happened in one flow. According to Wireshark-Statistics-Conversations (Figure 16), the connection between the phone and the controller is considered to be the longest (approximately 16 minutes) in the traffic. However, based on previous RATs analysis in the Android Mischief dataset, connections to services such as Facebook, Instagram, etc. might be longer than the 16 minutes of this malicious connection. Due to the victim reconnecting to the C&C several times after the connection was closed, Wireshark displays a number of flows to the C&C with a really short duration (Figure 17).

Address A	Port A	Address B	Port B	Duration
147.32.83.245	46032	147.32.83.157	1337	1008.9216
147.32.83.245	60762	172.217.23.206	443	459.8733
147.32.83.245	60752	104.244.42.194	443	363.4646
147.32.83.245	48924	216.58.201.78	443	300.0654
147.32.83.245	60640	104.244.42.194	443	192.8217
147.32.83.245	42290	209.237.199.128	443	191.4794
147.32.83.245	48952	13.83.65.43	443	82.9226
147.32.83.245	60742	172.217.23.206	443	60.0113
147.32.83.245	56804	104.244.42.3	443	40.7457
147.32.83.245	60832	104.244.42.194	443	40.0352

Figure 16. TOP connections from Wireshark-Statistics-Conversations sorted by the flow duration. The connection between the victim and C&C is the longest.

Address A	Port A	Address B	Port B	Duration
147.32.83.245	46032	147.32.83.157	1337	1008.9216
147.32.83.245	46052	147.32.83.157	1337	3.0035
147.32.83.245	46054	147.32.83.157	1337	3.0025
147.32.83.245	46050	147.32.83.157	1337	2.9998
147.32.83.245	46056	147.32.83.157	1337	1.0000

FIgure 17. Wireshark displays reconnections to the C&C as the flows of really short duration.

Conclusion

In this blog we have analyzed the network traffic from a phone infected with a unique command line AndroRAT. Due to the RAT simple communication protocol, we were able to decode its connection. The command line androRAT does not seem to be complex in its communication, however, it is quite sophisticated in its work. It is not interrupting throughout the whole communication compared to other RATs in the dataset.

To summarize, the details found in the network traffic of this RAT are:

- The C&C sends the packets in plaintext without any structure.
- The infected phone sends the packets in plaintext without any structure.
- The communication between the C&C and the phone is done in one flow of long duration (approximately 16 minutes).
- Even though the connection between the controller and the phone was closed, the phone tries to reconnect every 3 seconds.
- There is no heartbeat in the traffic between the phone and the controller.

Biographies


KAMILA BABAYEVA

Sebastian Garcia is a malware researcher and security teacher with experience in applied machine learning on network traffic. He founded the Stratosphere Lab, aiming to do impactful security research to help others using machine learning. He believes that free software and machine learning tools can help better protect users from abuse of our digital rights. He researches on machine learning for security, honeypots, malware traffic detection, social networks security detection, distributed scanning (dnmap), keystroke dynamics, fake news, Bluetooth analysis, privacy protection, intruder detection, and microphone detection with SDR (Salamandra). He co-founded the MatesLab hackspace in Argentina and co-founded the Independent Fund for Women in Tech. @eldracote. https://www.researchgate.net/profile/Sebastian_Garcia6

Kamila Babayeva is a 20 years old and third-year bachelor student in the Computer Science and Electrical Engineering program at the Czech Technical University in Prague. She is a researcher in the Civilsphere project, a project dedicated to protecting civil organizations and individuals from targeted attacks. Her research focuses on helping people and protecting their digital rights by developing free software based on machine learning. Initially, she worked as a junior Malware Reverser. Currently, Kamila leads the development of the Stratosphere Linux Intrusion Prevent System (Slips), which is used to protect the civil society in the Civilsphere lab.

SEBASTIAN GARCIA